



Yapay Zeka Yöntemleri

2021-2022 Bahar Yarıyılı

Proje 2 Raporu

19.06.2022

*Doğal Dil İşleme (Natural Language Processing)
ile*

Görüş / Duygu Analizi (Sentiment Analysis)

Bir konu hakkında gönderilen tweet'lerin olumlu mu olumsuz mu olduğunun bulunması

Mustafa Kaya - 05190000059

Veli Yaşar - 05190000841

Şükriye Züleyha Alkan - 05190000107

İçindekiler:

- 1) Problemin Tanımı
- 2) Araştırma/Ön Çalışma
- 3) Kullanılan Ortam, Yöntem ve Kütüphaneler
- 4) Kaynak Kodlar ve Açıklamaları
- 5) Ek 1: Başarı artırmak için kullanılan yöntemler ve artış miktarı
- 6) Ek 2: Daha önce yapılmış çalışmalar ile karşılaştırma
- 7) Ek 3: Kullanılan hazır kodlar/bağlantılar
- 8) Ek 4:
 - a) Kümeleme Algoritmaları için performans ölçütleri
 - b) Derin öğrenmede Transformer Model
- 9) İş Bölümü
- 10) Öz Değerlendirme Tablosu

0. NPL.ipnb'nin çalışması için gerekli .txt dosyası

İndirme Linki: [download \(677 MB\)](#)

1. Problemin Tanımı

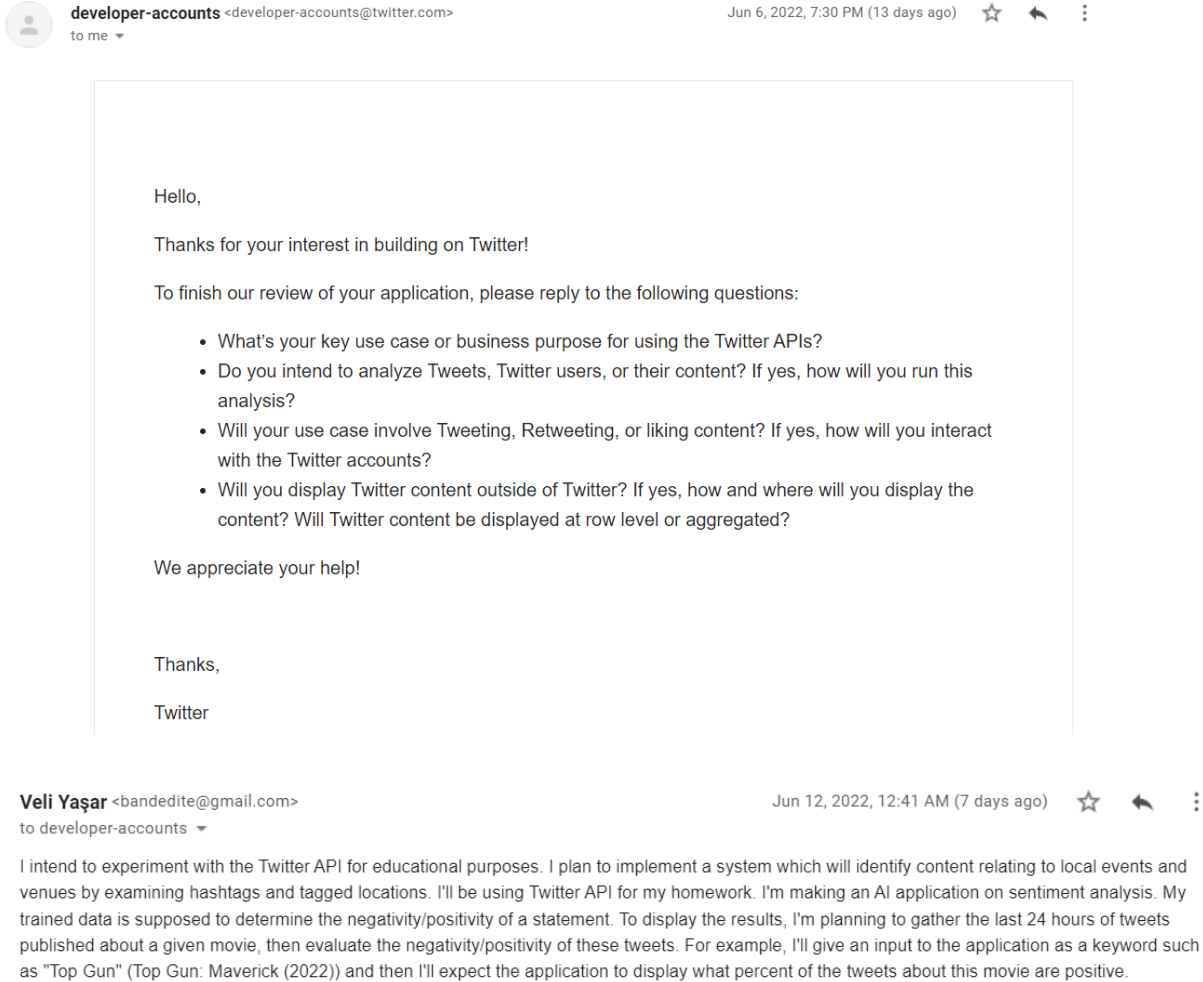
Projemizde verilen bir zaman aralığında bir konu/hashtag hakkındaki tweet'ler için duygu analizi yapmaya çalıştık. Konuyu "Lightyear"(2022) filmi, zaman aralığını ise son 24 saat olarak belirledik.

Projenin öneminden bahsetmek gerekirse; geliştirilen bir ürünün insanlar tarafından nasıl karşılandığını bilmek üretici için kritik geri bildirimler sağlar. Üretici bu sayede hatalarını fark edip daha iyisini hedefleyebilir. O yüzden çıkartılan ürünün kamuoyu tarafından nasıl karşılandığının analizinin yapılması kritik bir öneme sahiptir. Twitter da insanların en hızlı tepki verdiği platformlardan biri olarak duygu analizi için ciddi bir kaynak sağlıyor.

2. Araştırma/Ön Çalışma

Seçtiğimiz konu hakkında kendimizi geliştirmek için YouTube'dan birtakım eğitici videolar izledik. Algoritmaları araştırarak hangilerinin bize daha hızlı ve etkin bir şekilde sonuç verebileceğini araştırdık.

Elde ettiğimiz eğitimli sistemde kullanacağımız tweet'leri çekebilmek için Twitter API ile Twitter verisini kullanmamız gerekiyor. Twitter ilk aşamada verilere erişebilmemiz için developer hesabı açmamızı istiyor. Bunun için başvurularımızı yaptık. Twitter sadece birimize developer hesabı vermeyi onayladı.



3. Kullanılan Ortam, Yöntem ve Kütüphaneler

Bu projeyi Python, Anaconda, Jupyter Notebook teknolojilerini kullanarak geliştirdik.

İlk aşamada IMDB-Dataset.csv dosyasındaki yorumları kullanarak bir cümlenin olumlu ya da olumsuz olmasını tespit edecek sistemin eğitilmesini sağladık. Temel olarak veri setinin çekilmesi için BeautifulSoup algoritmasını, veri setinin eğitimi için Tensorflow/Keras kütüphanesini kullandık. Bunların yanında Tweepy kütüphanesiyle Twitter API'ya bağlanıp tweet'lere eriştik ve bu verilerin analizini yaptık.

4. Kaynak Kodlar ve Açıklamaları

Öncelikle *pandas* ve *numpy* kütüphaneleri import ettik ve sonrasında da bize verilen "imdb_dataset.csv" dosyasını okuduk.

```
import pandas as pd
import numpy as np
import warnings

dataset = pd.read_csv('IMDB Dataset.csv')
dataset
#warnings.filterwarnings('ignore')
```

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive
...
49995	I thought this movie did a down right good job...	positive
49996	Bad plot, bad dialogue, bad acting, idiotic di...	negative
49997	I am a Catholic taught in parochial elementary...	negative
49998	I'm going to have to disagree with the previou...	negative
49999	No one expects the Star Trek movies to be high...	negative

50000 rows x 2 columns

```
dataset.sentiment.value_counts()

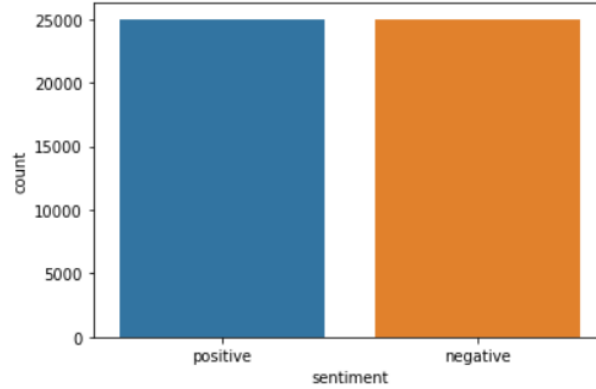
positive    25000
negative    25000
Name: sentiment, dtype: int64
```

Okuduğumuz dosyamız ile alaklı datasetlerini okuyup bunların değerlerini ve sayılarını yazdırdık ve output olarak bizlere 25000 pozitif ve 25000 negatif değer döndürdü.

Sonrasında *seaborn* kütüphanesinden faydalanarak bizlere grafiksel olarak pozitif ve negatif verilerinin adedini bu şekilde gösterdim.

```
import seaborn as sns
sns.countplot(dataset.sentiment)
```

<AxesSubplot:xlabel='sentiment', ylabel='count'>



BeautifulSoup algoritmalarından faydalanarak textlerdeki gereksiz ifadeleri sildik. Bunu *cleantext* fonksiyonu ile yaptık. Sonrasında da output olarak tekrardan bizlere 50000 adet veri geri döndü.

```
from bs4 import BeautifulSoup
import re
def cleanText(text):
    text = BeautifulSoup(text, "lxml").text
    text = re.sub(r'\\|\\|\\|', r' ', text)
    text = re.sub(r'http\S+', r'<URL>', text)
    text = text.lower()
    text = text.replace('x', '')
    return text
dataset['review'] = dataset['review'].apply(cleanText)
dataset
```

	review	sentiment
0	one of the other reviewers has mentioned that ...	positive
1	a wonderful little production. the filming tec...	positive
2	i thought this was a wonderful way to spend ti...	positive
3	basically there's a family where a little boy ...	negative
4	petter matter's "love in the time of money" is...	positive
...
49995	i thought this movie did a down right good job...	positive
49996	bad plot, bad dialogue, bad acting, idiotic di...	negative
49997	i am a catholic taught in parochial elementary...	negative
49998	i'm going to have to disagree with the previou...	negative
49999	no one epects the star trek movies to be high ...	negative

50000 rows × 2 columns

```

corpus = []
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize
for text in dataset['review']:
    words = [word.lower() for word in word_tokenize(text)]
    corpus.append(words)

num_words = len(corpus)

```

```

[nltk_data] Downloading package punkt to
[nltk_data] C:\Users\bande\AppData\Roaming\nltk_data...
[nltk_data] Package punkt is already up-to-date!

```

Burada yapılan işlemler genel itibari ile önce cümleleri tokenize işlemi ile kelimeler haline getirmektedir. Sonrasında verisetimizi train etmeye başlıyoruz. Bir sonraki kısımda artık ve işleyişini göreceğiz.

```

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

le = LabelEncoder()
training_reviews, testing_reviews, training_labels, testing_labels = train_test_split(dataset['review'].values, dataset['sentiment'],
training_labels = le.fit_transform(training_labels)
testing_labels = le.fit_transform(testing_labels)

```

```

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

tokenizer = Tokenizer(num_words=10000, oov_token='<OOV>')
tokenizer.fit_on_texts(training_reviews)
word_index = tokenizer.word_index
training_sequence = tokenizer.texts_to_sequences(training_reviews)
testing_sequence = tokenizer.texts_to_sequences(testing_reviews)
train_pad_sequence = pad_sequences(training_sequence, maxlen = 200, truncating= 'post', padding = 'pre')
test_pad_sequence = pad_sequences(testing_sequence, maxlen = 200, truncating= 'post', padding = 'pre')
print('Found unique tokens: {}'.format(len(word_index)))

```

Found unique tokens: 113808

```

embedded_words = {}
with open('glove.6B.200d.txt', encoding="utf8") as file:
    for line in file:
        words, coeff = line.split(maxsplit=1)
        coeff = np.array(coeff.split(), dtype = float)
        embedded_words[words] = coeff

```

```

embedding_matrix = np.zeros((len(word_index) + 1, 200))
for word, i in word_index.items():
    embedding_vector = embedded_words.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector

```

Sequentiallar'dan faydalanarak burada modelimizi oluşturmuş bulunuyoruz.

```
: import tensorflow as tf

model = tf.keras.Sequential([tf.keras.layers.Embedding(len(word_index) + 1,200,weights=[embedding_matrix],input_length=200,
trainable=False),
tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),
tf.keras.layers.Dropout(0.5),
tf.keras.layers.Dense(256,activation = 'relu',),
tf.keras.layers.Dense(128,activation = 'relu'),
tf.keras.layers.Dropout(0.5),
tf.keras.layers.Dense(1,activation = tf.nn.sigmoid))

model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
embedding (Embedding)	(None, 200, 200)	22761800
bidirectional (BidirectionalL)	(None, 128)	135680
dropout (Dropout)	(None, 128)	0
dense (Dense)	(None, 256)	33024
dense_1 (Dense)	(None, 128)	32896
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 1)	129
=====		
Total params: 22,963,529		
Trainable params: 201,729		
Non-trainable params: 22,761,800		

Burada verilerimiz eğitime başladı ve görüldüğü üzere “Epoch 1/30” deki accuracy = 0.7573 ve validation accuracy = 0.8338 fakat ilerleyen “Epoch”larımızda verilerimizdeki artış başlamaktadır. Verilerimizdeki doğruluk oranları da artmaya başlamaktadır.

```
model.compile(loss = tf.keras.losses.BinaryCrossentropy() , optimizer='Adam' , metrics = 'accuracy')
history = model.fit(train_pad_sequence,training_labels,epochs = 30 ,validation_data=(test_pad_sequence,testing_labels))

Epoch 1/30
1250/1250 [=====] - 142s 111ms/step - loss: 0.4990 - accuracy: 0.7487 - val_loss: 0.3828 - val_accuracy: 0.8362
Epoch 2/30
1250/1250 [=====] - 125s 100ms/step - loss: 0.3660 - accuracy: 0.8451 - val_loss: 0.3311 - val_accuracy: 0.8571
Epoch 3/30
1250/1250 [=====] - 124s 100ms/step - loss: 0.3307 - accuracy: 0.8592 - val_loss: 0.3345 - val_accuracy: 0.8601
Epoch 4/30
1250/1250 [=====] - 129s 103ms/step - loss: 0.3014 - accuracy: 0.8739 - val_loss: 0.3057 - val_accuracy: 0.8716
Epoch 5/30
1250/1250 [=====] - 126s 100ms/step - loss: 0.2789 - accuracy: 0.8845 - val_loss: 0.3042 - val_accuracy: 0.8729
Epoch 6/30
1250/1250 [=====] - 126s 101ms/step - loss: 0.2568 - accuracy: 0.8954 - val_loss: 0.2974 - val_accuracy: 0.8722
Epoch 7/30
1250/1250 [=====] - 125s 100ms/step - loss: 0.2343 - accuracy: 0.9038 - val_loss: 0.3129 - val_accuracy: 0.8741
Epoch 8/30
1250/1250 [=====] - 125s 100ms/step - loss: 0.2146 - accuracy: 0.9150 - val_loss: 0.3274 - val_accuracy: 0.8760
Epoch 9/30
1250/1250 [=====] - 126s 101ms/step - loss: 0.1924 - accuracy: 0.9249 - val_loss: 0.3245 - val_accuracy: 0.8733
Epoch 10/30
1250/1250 [=====] - 131s 104ms/step - loss: 0.1750 - accuracy: 0.9327 - val_loss: 0.3413 - val_accuracy: 0.8624
Epoch 11/30
1250/1250 [=====] - 141s 112ms/step - loss: 0.1557 - accuracy: 0.9414 - val_loss: 0.3475 - val_accuracy: 0.8607
Epoch 12/30
1250/1250 [=====] - 138s 111ms/step - loss: 0.1408 - accuracy: 0.9456 - val_loss: 0.3966 - val_accuracy: 0.8636
```

...


```
Epoch 26/30
1250/1250 [=====] - 127s 102ms/step - loss: 0.0424 - accuracy: 0.9852 - val_loss: 0.7215 - val_accu
racy: 0.8661
Epoch 27/30
1250/1250 [=====] - 124s 99ms/step - loss: 0.0383 - accuracy: 0.9869 - val_loss: 0.8118 - val_accu
racy: 0.8659
Epoch 28/30
1250/1250 [=====] - 124s 99ms/step - loss: 0.0439 - accuracy: 0.9853 - val_loss: 0.6999 - val_accu
racy: 0.8615
Epoch 29/30
1250/1250 [=====] - 131s 105ms/step - loss: 0.0350 - accuracy: 0.9885 - val_loss: 0.7259 - val_accu
racy: 0.8590
Epoch 30/30
1250/1250 [=====] - 132s 106ms/step - loss: 0.0345 - accuracy: 0.9885 - val_loss: 0.8327 - val_accu
racy: 0.8615
```

Burada veri setimizin sonunu görüyoruz.

Sonrasında yapılan işlemlerle *matplotlib* kütüphanesi yardımı ile oluşan sonuçları grafiklerde göstermekteyiz.

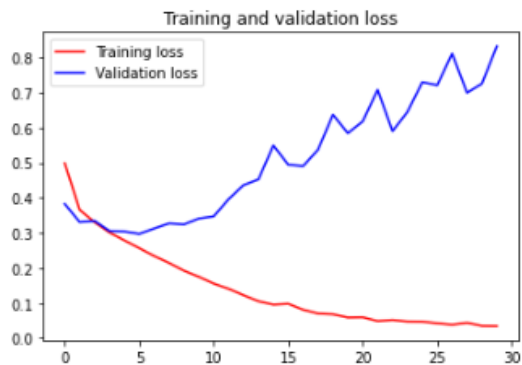
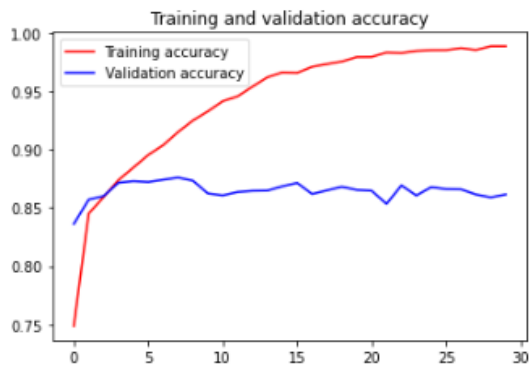
```
import matplotlib.pyplot as plt

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(acc))

plt.plot(epochs, acc, 'r', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend(loc=0)
plt.figure()

plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend(loc=0)

plt.show()
```



Bu kısımda yaptığımız işlemle sonuçları grafiklerde gösterdikten sonra, “training accuracy” ve “validation accuracy” kısımları ile alakalı bize dönen verilerde maksimum değerleri yazdırdık.

```
: print('Training Accuracy: {}'.format(max(acc)))
   print('Validation Accuracy: {}'.format(max(val_acc)))
```

```
Training Accuracy: 0.9885249733924866
Validation Accuracy: 0.8759999871253967
```

Bu kısımda ise bir “get_result” metodu yazdık, bu metoda bir cümle göndereceğiz ve bize bu cümlelerin pozitif ya da negatif mi olduğunu söyleyecek. Örnek olarak ise “I can watch this movie forever just because of the beauty in its cinematography.” cümlesini verdik ve *get_result* metoduna gönderdiğimizde bu text’i bize “positive” olarak bir sonuç döndü.

```
def get_result(txt):
    seq = tokenizer.texts_to_sequences(txt)
    padded = pad_sequences(seq, maxlen=200, dtype='int32', value=0)
    pred = model.predict(padded)
    print(pred)
    labels = [0,1]
    result = labels[round(pred[0][0])]
    result_dict = {0: 'negative', 1: 'positive'}
    print(result)
    return result_dict[result]
```

```
m = "This movie was not good at all. It had some good parts like the acting was pretty good but the story was not impressing at all"
get_result([m])
```

```
1/1 [=====] - 1s 1s/step
[[0.00181467]]
0
'negative'
```

```
m = "I can watch this movie forever just because of the beauty in its cinematography."
get_result([m])
```

```
1/1 [=====] - 0s 28ms/step
[[0.9372816]]
1
'positive'
```

Proje için Twitter aracılığı ile verilen bir key ya da hashtag sonrasında o kelime ile alakalı son 1 gün içerisinde atılmış bütün tweetleri döndürüp sonrasında bunların ne kadarı pozitif ve negatif olduğunun analizini bizlere sunmaktadır.

Bunun için öncelikle Twitter developer hesabı aldık ve Twitter tarafından api ye bağlanmak için bizlere bir takım “developer key”ler verildi ve bu “key”ler aracılığı ile erişim sağladık.

Bir *percentage* fonksiyonu oluşturduk. Bunun amacı projenin en sonunda kullanacağımız grafiği görüntülemek.

isEnglish fonksiyonu ise twitlerin ingilizce olup olmadığını anlamak içindir.

Sonrasında ise 24 saat içerisinde atılan tweetleri bulmak için *datetime* ve *timedelta* classlarını import ediyoruz.

```
import tweepy as tw

consumer_key = 'qOV8d9cSjdHYJeLrj19VPhmdt '
consumer_secret = 'vpkxcxwMo859ZrK7RMs8s7rYv68yabsByQ9s2o3ohY68ewHj4s'
access_token = '1058720835568128000-CwnGkKH1dkxhy9Hnf35RzVGZ2qnNjd'
access_token_secret = 'ua00JvIq4nkZng0xqtRpkJHf6dQK0R3smK4nF6I4G2020'
bearer_token = "AAAAAAAAAAAAAAAAAAAAALdgEAAAAAcqfFwbU6d3dcjZdix0LXufZUBBs%3DrQ8aT9ewhYZ1ETk60JYX0kgne"
```

```
try:
    auth = tw.OAuth2BearerHandler(bearer_token)
    api = tw.API(auth)

except:
    print("Error: Authentication Failed")
```

```
def percentage(part,whole):
    return 100 * float(part)/float(whole)
```

```
def isEnglish(text):
    try:
        text.encode(encoding='utf-8').decode('ascii')
    except UnicodeDecodeError:
        return False
    else:
        return True
```

```
from datetime import datetime, timedelta

now = datetime.today().now()
prev=now-timedelta(days=1)
now=now.strftime("%Y-%m-%d")
prev=prev.strftime("%Y-%m-%d")
prev
```

'2022-06-18'

Tekrardan cümleleri tokenize ederek kelime haline dönüştürüyoruz ve bütün cümleyi *lower* ile küçültüyoruz.

```
import nltk
nltk.download('stopwords')
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer

stop_words = set(stopwords.words('english'))

def preprocess_text(text):

    text.lower()
    # Remove urls
    text = re.sub(r"http\S+|www\S+|https\S+", '', text, flags=re.MULTILINE)
    # Remove user @ references and '#' from tweet
    text = re.sub(r'\@w+|\#','', text)
    # Remove punctuations
    #text = text.translate(str.maketrans('', '', string.punctuation))
    # Remove stopwords
    text_tokens = word_tokenize(text)
    filtered_words = [w for w in text_tokens if not w in stop_words]

    #ps = PorterStemmer()
    #stemmed_words = [ps.stem(w) for w in filtered_words]
    #Lemmatizer = WordNetLemmatizer()
    #lemma_words = [lemmatizer.lemmatize(w, pos='a') for w in stemmed_words]

    return " ".join(filtered_words)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data] C:\Users\bande\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!
```

“get_tweets” fonksiyonu oluşturuyoruz ve sonrasında da text’leri çekip sırasıyla yukarıda belirtilen fonksiyonlara gönderiyoruz. Gerekli işlemler yapıldıktan sonra da işimize yarayan verileri kendi dizimize aktarıyoruz.

```
def get_tweets(query, count):
    tweets = []
    query = query + " -filter:retweets"

    try:
        fetched_tweets = tw.Cursor(api.search_tweets,
                                   q = query,
                                   lang = 'en',
                                   since = prev,
                                   until = now).items(count)
        #api.search(q = query, count = count, lang = 'en')

        for tweet in fetched_tweets:
            parsed_tweet = {}
            print(tweet.text)
            if isEnglish(tweet.text) == True:
                parsed_tweet['text'] = preprocess_text(tweet.text)
                parsed_tweet['sentiment'] = get_result([preprocess_text(tweet.text)])

                if tweet.retweet_count > 0:
                    if parsed_tweet not in tweets:
                        tweets.append(parsed_tweet)

                else:
                    tweets.append(parsed_tweet)

        return tweets

    except AttributeError as e:
        print("Error : " + str(e))
```

```
key = "#" + input ("Enter key or hashtag to search about: ")
numberOfTweets = int(input("Enter how many tweets to analyze: "))
```

```
Enter key or hashtag to search about: Lightyear
Enter how many tweets to analyze: 100
```

Sonrasındaki kısımda ise pozitif ve negatif tweetleri bir diziye atıp “Lightyear” filmi ile alakalı belirtilen parametrelere göre gelen tweetler aşağıda gösterilmektedir.

```
tweets = get_tweets(key, numberOfTweets)
pzt_tweets = [tweet for tweet in tweets if tweet['sentiment'] == 'positive']
neg_tweets = [tweet for tweet in tweets if tweet['sentiment'] == 'negative']
```

Unexpected parameter: since

Do I see #Lightyear in theaters or wait for it to hit Dizz Dizz Plus?

1/1 [=====] - 0s 31ms/step

[[0.94481516]]

1

Today I'm going to see #Lightyear <https://t.co/qCCk8U6OVn>

1/1 [=====] - 0s 25ms/step

[[0.99867254]]



1

Lightyear is a better version of interstellar #Lightyear

1/1 [=====] - 0s 26ms/step

[[0.76340145]]

1

#Lightyear was so good ahhhhh my heart  

My week is free. Anyone want to see the new #Lightyear movie with me.

1/1 [=====] - 0s 28ms/step

[[0.9788121]]

1

"I'm supposed to be like my grandma"

'No, you're Izzy!' shouts a little girl from the audience. #Lightyear

1/1 [=====] - 0s 30ms/step

[[0.9178582]]

1

#Lightyear To infinity....and.... BEYOND <https://t.co/hCIKG8Ncwt>

How does Andy watch #Lightyear and not buy Sox? <https://t.co/nrbdbFKItU>

1/1 [=====] - 0s 25ms/step

[[0.8917448]]

1

I think Luke would agree his first movie theater experience was a 10/10! #Lightyear <https://t.co/SOX1fUkksj>

1/1 [=====] - 0s 26ms/step

[[0.9990864]]

1

Burada pozitif ve negatif array'lerle atılan tweetlerin bir nevi sağlamasını yapmaktayız.

Sonrasında bu tweetler ile alakalı pozitif ve negatiflik yüzdelik oranları yazdırılmıştır.

pzt_tweets

```
[{'text': 'Do I see Lightyear theaters wait hit Dizz Dizz Plus ?',  
  'sentiment': 'positive'},  
{'text': 'Today I 'm going see Lightyear', 'sentiment': 'positive'},  
{'text': 'Lightyear better version interstellar Lightyear',  
  'sentiment': 'positive'},  
{'text': 'My week free . Anyone want see new Lightyear movie .',  
  'sentiment': 'positive'},  
{'text': '"" I 'm supposed like grandma "" 'No , 're Izzy ! ' shouts little girl audience . Lightyear',  
  'sentiment': 'positive'},  
{'text': 'How Andy watch Lightyear buy Sox ?', 'sentiment': 'positive'},  
{'text': 'I think Luke would agree first movie theater experience 10/10 ! Lightyear',  
  'sentiment': 'positive'},  
{'text': 'Seated Lightyear excited', 'sentiment': 'positive'},  
{'text': 'The cat I ever want Sox Lightyear', 'sentiment': 'positive'},  
{'text': 'Lightyear spoiler context :', 'sentiment': 'positive'},  
{'text': 'Ca n't wait see Lightyear tomorrow !', 'sentiment': 'positive'},  
{'text': 'What 's favorite Pixar movie time ? FilmTwitter ToyStory Lightyear Pixar',  
  'sentiment': 'positive'},  
{'text': 'Saw Lightyear Made drawing .', 'sentiment': 'positive'},  
{'text': 'Enjoyed lightyear memorable Pixar film good summer romp .',  
  'sentiment': 'positive'},  
{'text': 'Lightyear Review via', 'sentiment': 'positive'},  
{'text': 'Lightyear fam .', 'sentiment': 'positive'},  
{'text': 'About see Lightyear . Will see turns .', 'sentiment': 'positive'},  
{'text': 'Lightyear reimagined PS1 game brand new video .',  
  'sentiment': 'positive'},  
{'text': 'Me coming way make Lightyear sequel tie Buzz Lightyear Star Command',  
  'sentiment': 'positive'},  
{'text': 'Met new Mr. Lightyear Tomorrowland ! Disneyland ToInfinityAndBeyond',  
  'sentiment': 'positive'},  
{'text': 'A great film & amp ; vastly superior Lightyear',  
  'sentiment': 'positive'},  
{'text': 'Happy Pride BuzzLightyear Lightyear', 'sentiment': 'positive'},  
{'text': 'Worth good 11 years ! ToyStory Lightyear', 'sentiment': 'positive'},  
{'text': 'Taika Waititi , Keke Palmer Chis Evans light blue carpet Lightyear premiere',  
  'sentiment': 'positive'},  
{'text': 'Movie Review : Lightyear', 'sentiment': 'positive'},  
{'text': 'Thank 8.5 / 10 Lightyear', 'sentiment': 'positive'},  
{'text': 'My review Lightyear starring', 'sentiment': 'positive'},  
{'text': 'My thoughts Lightyear', 'sentiment': 'positive'},  
{'text': 'Double feature Saturday AMC Lightyear TopGunMaverick',  
  'sentiment': 'positive'},  
{'text': 'To infinity ... Lightyear', 'sentiment': 'positive'}]
```

neg_tweets

```
[{'text': 'Saw tio Pixar film . Why I lowk tearin Lightyear',  
  'sentiment': 'negative'},  
{'text': 'PIXAR . Lightyear PLEASE GIVE US A WOODYS ROUND UP MOVIE JUST A SILLY COWBOY MOVIE P L E A S E',  
  'sentiment': 'negative'},  
{'text': 'rolling Lightyear im sorry say shoutout buzzheads tho',  
  'sentiment': 'negative'},  
{'text': 'Yes I adult spent money Buzz Lightyear action figure seeing movie .',  
  'sentiment': 'negative'},  
{'text': 'Lightyear sad , strange little movie , pity',  
  'sentiment': 'negative'},  
{'text': 'Snuck party good mirrors Lightyear', 'sentiment': 'negative'},  
{'text': 'Looks like pedo going miss Lightyear tonight , maybe decade . DisneyGroomer Uniform .',  
  'sentiment': 'negative'},  
{'text': 'watching Lightyear', 'sentiment': 'negative'},  
{'text': 'Lightyear ( 2022 ) Lightyear 2022Movies Poster Animation Pixar',  
  'sentiment': 'negative'},  
{'text': 'Need watching Lightyear', 'sentiment': 'negative'},  
{'text': 'The Pizza Planet truck made appearance Lightyear',  
  'sentiment': 'negative'}]
```

```
print("Positive tweets percentage: {} %".format(percentage(len(pzt_tweets),len(tweets))))  
print("Negative tweets percentage: {} %".format(percentage(len(neg_tweets),len(tweets))))
```

Positive tweets percentage: 73.17073170731707 %
Negative tweets percentage: 26.829268292682926 %


```
positive_p = percentage(len(pzt_tweets),len(tweets))
negative_p = percentage(len(neg_tweets),len(tweets))
```

```
positive_p = format(positive_p, '.2f')
negative_p = format(negative_p, '.2f')
```

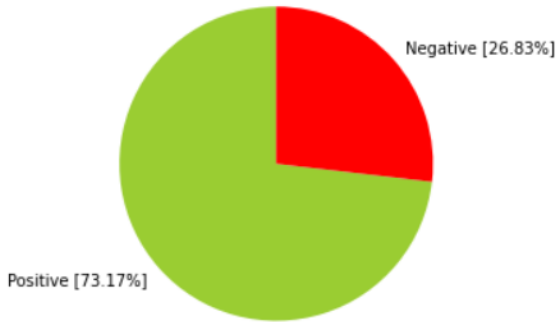
```
print("Positive tweets percentage: %" + positive_p)
print("Negative tweets percentage: %" + negative_p)
```

```
Positive tweets percentage: %73.17
Negative tweets percentage: %26.83
```

```
labels = ['Positive [' + str(positive_p) + '%]', 'Negative [' + str(negative_p) + '%]']
sizes = [positive_p, negative_p]
colors = ['yellowgreen', 'red']
fig, ax = plt.subplots()
ax.pie(sizes, labels=labels, startangle=90, colors=colors)
plt.title(key + " " + "Analysis of tweets about search movies: ")
ax.axis('equal')

plt.show()
```

#Lightyear Analysis of tweets about search movies:



Ve son olarak, yüzdeler kısımlarını düzenlemek adına, noktadan sonra 2 basamak alınacak şekilde sonuçlar yazdırıldı ve bu oranlara göre de bir pasta grafiği şekilde oluşturuldu.

5. Ek 1: Başarı artırmak için kullanılan yöntemler

Başarımı arttırmak için farklı algoritmalar kullanmayı denedim ve en verimli sonucu BeautifulSoup algoritması ile yakaladık.

6. Ek 2: Daha önce yapılmış çalışmalar ile karşılaştırma

[Sentiment Analysis of Twitter Texts Using Machine Learning Algorithms, Sakarya University](#)

Türkiye’de benzer bir çalışma yapıp yayınlanmış, kaynak kodları olmadan. Bu çalışma, kapsamının genişliği ve yöntemleri sebebiyle bize fazla yardımcı olamadı. Fakat projemizle karşılaştırmak gerekirse, bu çalışmada Support Vector Machines, Random Forest ve Gaussian Naive Bayes kullanılıyor, yorumlar pozitif/negatif/nötr olarak ayrılıyor, çok daha kapsamlı bir veri temizliği ve hazırlığı sağlanıyor, sistemi eğitmek için farklı bir “Trump’s tweets” adında bir veriseti kullanılıyor.

7. Ek 3: Kullanılan hazır kodlar/bağlantılar

Bütün kelimeleri ayırıp ve içerisindeki olumlu olumsuz örneklemeleri çıkarmak için internette daha hızlı sonuç veren algoritmaları araştırdık ve bunları da kendi koduma uyacak şekilde implemente ettik. Bu kod bloklarındaki gereksiz yerleri çıkardık ve kendi kod bloğumuza uygun şekilde düzenlemeler yaptık. Bunun yanında, farklılıklarımız aslında fonksiyonlarda yeterli bir şekilde hızlı ve işimi görecektir şekilde çalışmalarıydı. Kullandığımız çoğu fonksiyon bizim tarafımızdan yazıldı.

8. Ek 4:

a) Kümeleme Algoritmaları İçin Kullanılan Performans Ölçütleri

Silhouette Score

Siluet Puanı ve Siluet Grafiği, kümeler arasındaki ayırma mesafesini ölçmek için kullanılır. Bir kümedeki her noktanın komşu kümelerdeki noktalara ne kadar yakın olduğunun bir ölçüsünü gösterir. Bu ölçü $[-1, 1]$ aralığına sahiptir ve kümeler içindeki benzerlikleri ve kümeler arasındaki farklılıkları görsel olarak incelemek için kullanılan bir araçtır.

Rand Index

Bir kümedeki verinin değerini hesaplarken o veriyi aynı kümede olan diğer verilerle ikili olarak karşılaştırır. Bu işlemi aynı şekilde gerçek sınıf etiketleri için de yapar. Sonrasında bu yapılan işlemi ikili olarak karşılaştırır. Aynı mı yoksa farklı mı olduğunu bir tabloda tutar. Bu tablo üzerinden RI değerini hesaplar.

b) Derin Öğrenmede Transformer Model Nedir?

Transformer, giriş verilerinin her bir bölümünün önemini farklı şekilde ağırlıklandıran, kendi kendine dikkat mekanizmasını benimseyen bir derin öğrenme modelidir. Yapay zekâ (YZ) modellerinin girdilerinin belli kısımlarına seçici olarak odaklanmasını ve böylece daha hızlı ve etkin bir öğrenme amaçlanmıştır. Doğal dil işleme, bilgisayarla görme, konuşma tanıma, sembolik matematik ve pekiştirmeli öğrenme gibi çeşitli görevlerde kullanılmaktadır.

9. İş Bölümü

Mustafa Kaya: Ek 4, *preprocess_text()*, *get_result()*, debugging

Veli Yaşar: Rapor yazımı, Ek 2, Ek 3, *tweepy(Twitter API)*, *get_tweets()*

Şükriye Züleyha Alkan: Model için IMDB veriseti temini, TensorFlow/Keras, BeautifulSoup, *matplotlib* ile görüntüleme

10. Öz Değerlendirme Tablosu

	İstenen Madde	Var	Açıklama	Tahmini Not
1	Kapak Sayfası, Problemin Tanımı, Kullanılan Ortam, Yöntem ve Kütüphaneler, Araştırma (10)	+	Yapıldı	10
2	Önerilen Yöntem (10)	+	Yapıldı	10
3	Deneysel Çalışmalar (10)	+	Yapıldı	10
4	Proje Rapor Biçimi, Organizasyonu, Boyutu, Kalitesi, Kaynakça ve atıflar (10)	+	Yapıldı	10
5	Sonuç (10)	+	Yapıldı. Kodlar açıklandı.	10
6	Ek 1: Başarım İyileştirme (10)	-	Yapıldı fakat sonuçları kaydetmeyi unuttuğumuz için accuracy değerlerini paylaşamadık.	2
7	Ek 2 (10)	+	Yapıldı	10
8	Ek 3 (10)	+	Yapıldı	10
9	Ek 4 (10): Her madde 5'er puan	+	Yapıldı	10
10	Özdeğerlendirme Tablosu (10)	+	Yapıldı	10
100 üzerinde Toplam Not:				92